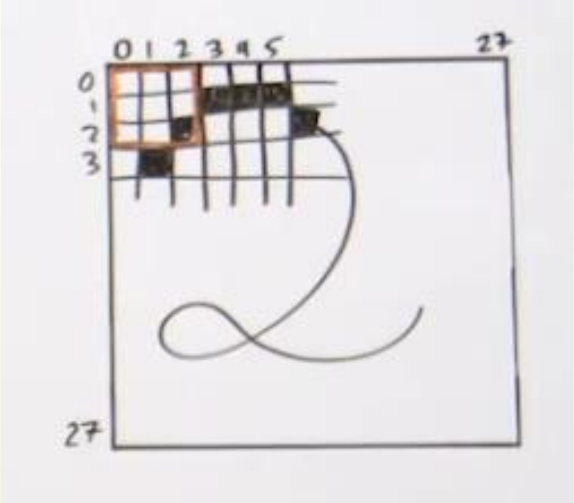
Deep Learning with CUDA
Dedicated architectures – CNN

Tomasz Szumlak

WFiIS AGH
27/03/2024, Kraków

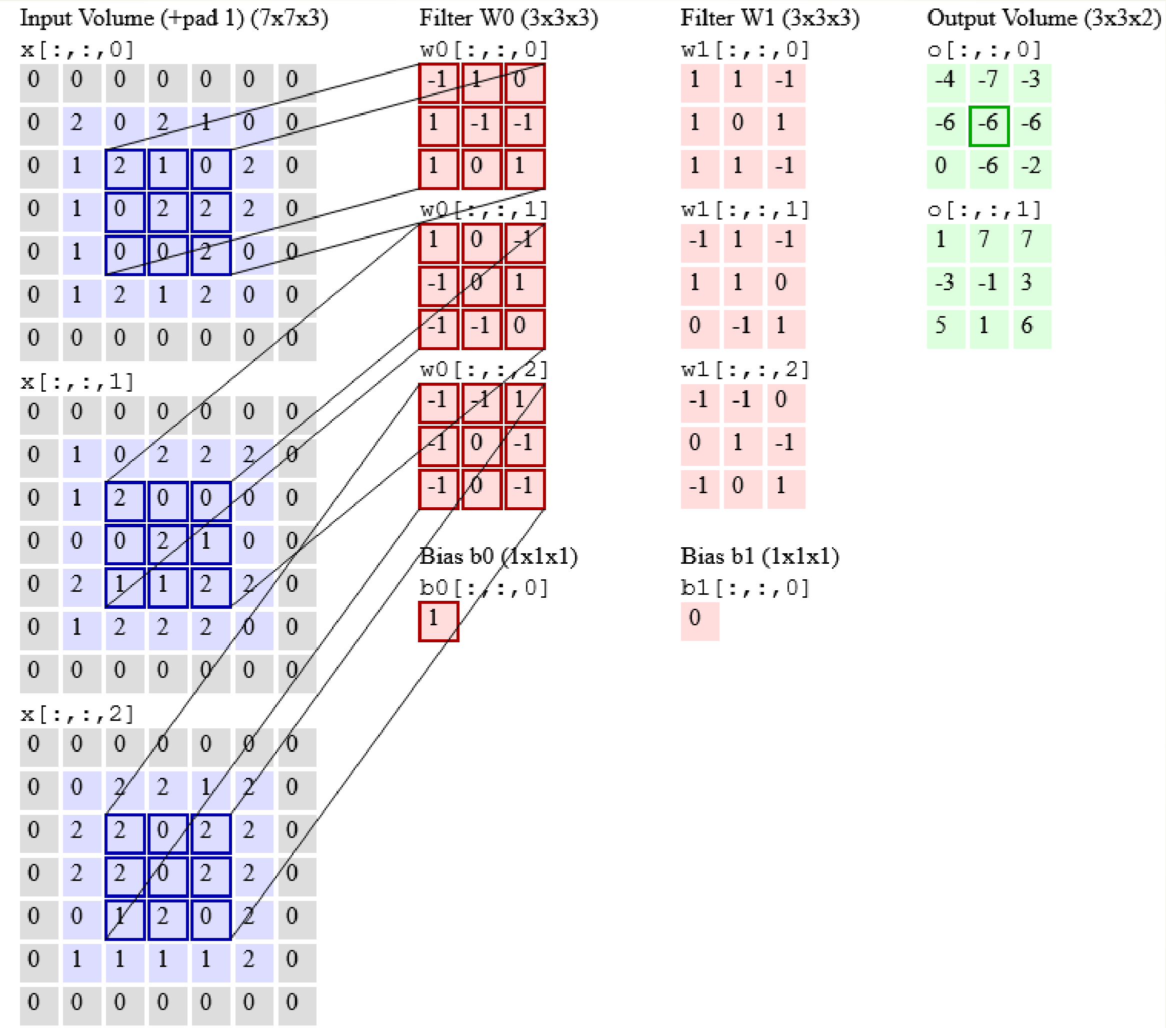
CNN – computer vision master

- One of the most sophisticated ANN models to date is the CNNs or Convolutional Neural Networks
- The main difference between the general dense model, which we have considered so far, **and CNN is specialisation** – enter the Conv layers
- Conv layers are strongly motivated by the biological retina



3

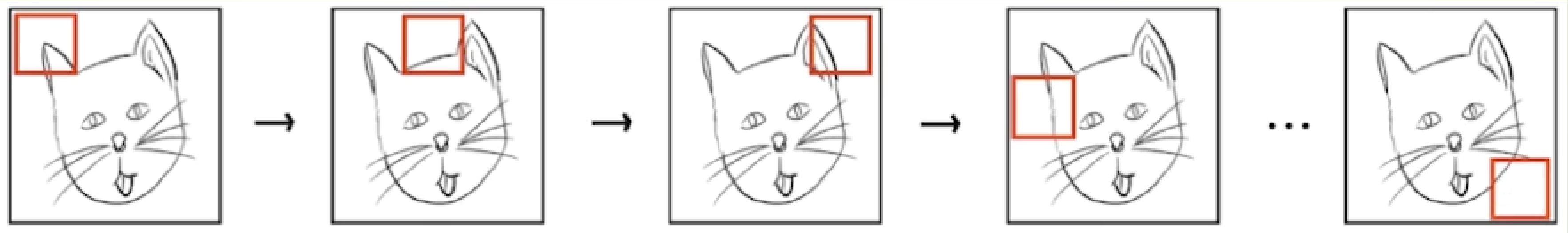
Conv layer demo…



https://cs231n.github.io/convolutional-networks/

Conv layer properties

- A few things to note…
  - We can process **colourful data** (R, G, B channels)
  - There is **extra padding** introduced around the original picture to get the equal attention of the scanning kernel to each pixel
  - **Kernels are a bit of an analogue of „advanced" neurons** (kernel parameters are trainable, and they are considered to be model weights)
  - For the example above, one can consider the kernel to be a 3D object with a single bias weight



Conv layer properties

- A few things to note…
  - The size of the kernel and the way it moves across the network architect can define the input data
  - The pace of the movement is called the stride (could be 1, 2…)
  - In our example, we have R-, B- and G-filters representing **one super neuron**
  - At each scan point, we **calculate the Hadamard product** (that is just an analogue of the grand neuron equation!!)
  - The total output of the first kernel filter is just a sum of all products plus the bias
  - In this way, **we create a compressed representation** of the input data
  - And we can have multiple kernels in each layer

Conv layer properties

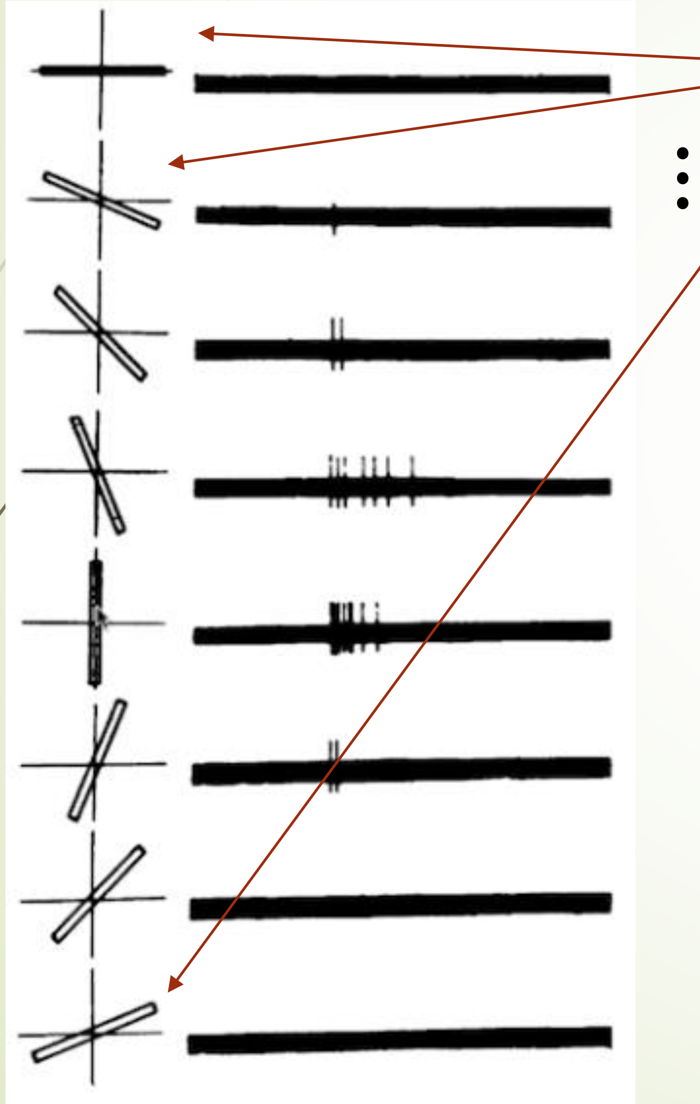
- Kernels represent some spatial features we are looking for
- On deeper layers, the features get more and more abstract
- The point is that in the output, the **significant positive** value will indicate the **presence of the feature**, and the **significant negative** will say that there is a **lack of one**

- **Important statement – Conv layers are designed to detect pattern/spatial features in position invariant way**
- The layers are faithful to the spatial structure of an input image
- This makes them robust against moving the picture, resizing, etc.

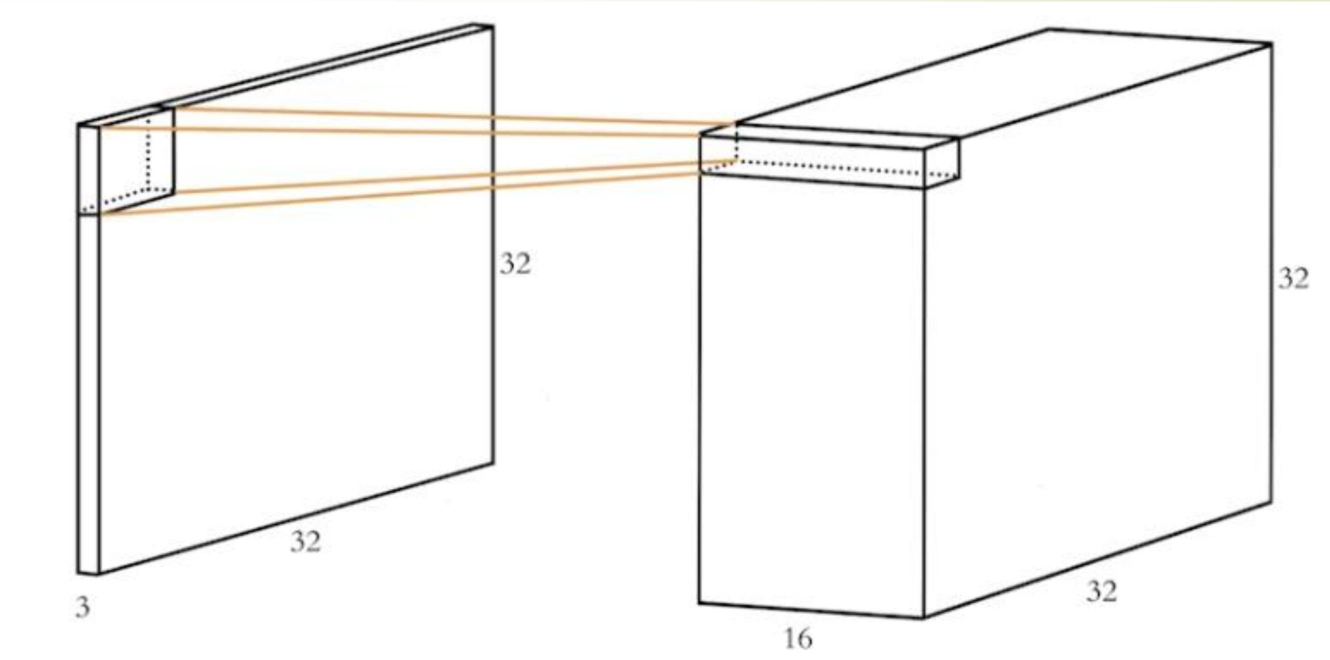
Conv layer properties

- Each layer can have many filters/kernels
- Each of which makes a representation of the input data
- If we compare the regular dense networks with CNNs, we see that the number of parameters required by CNNs are much, much lower
- On top of that, CNNs are going to be superior to any dense architecture in computer vision applications

Conv layer properties

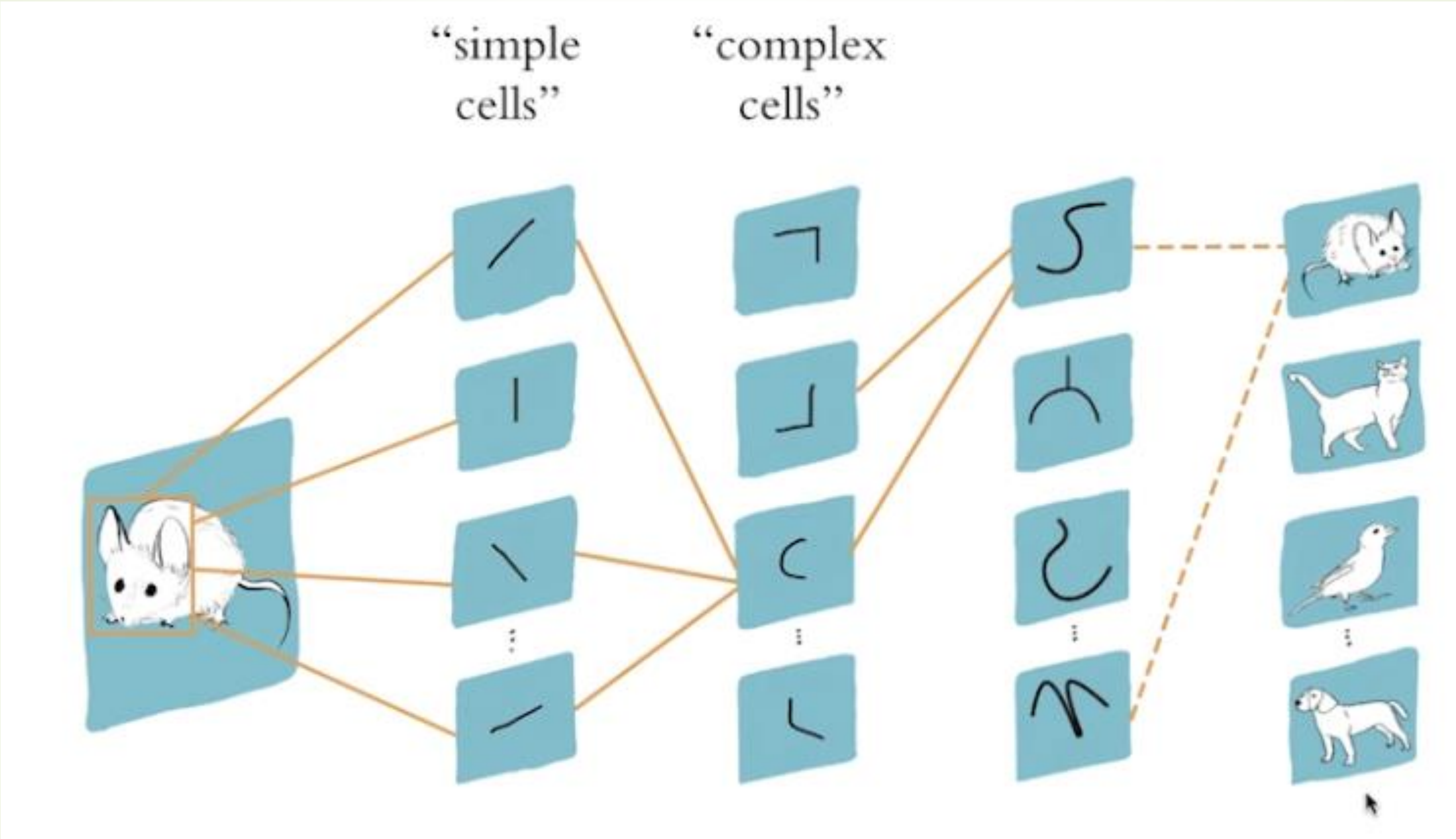


Each feature – one kernel

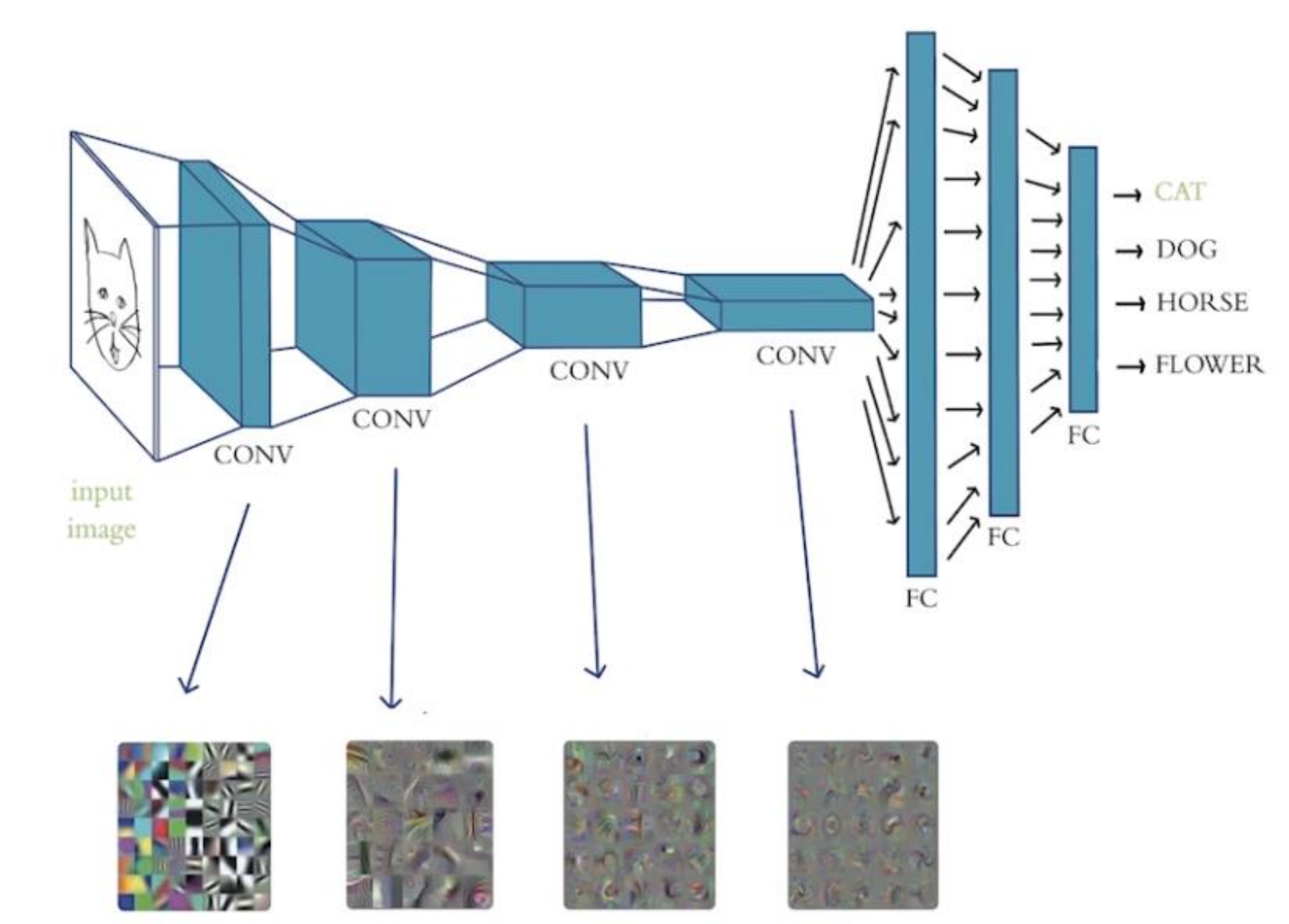


CNN

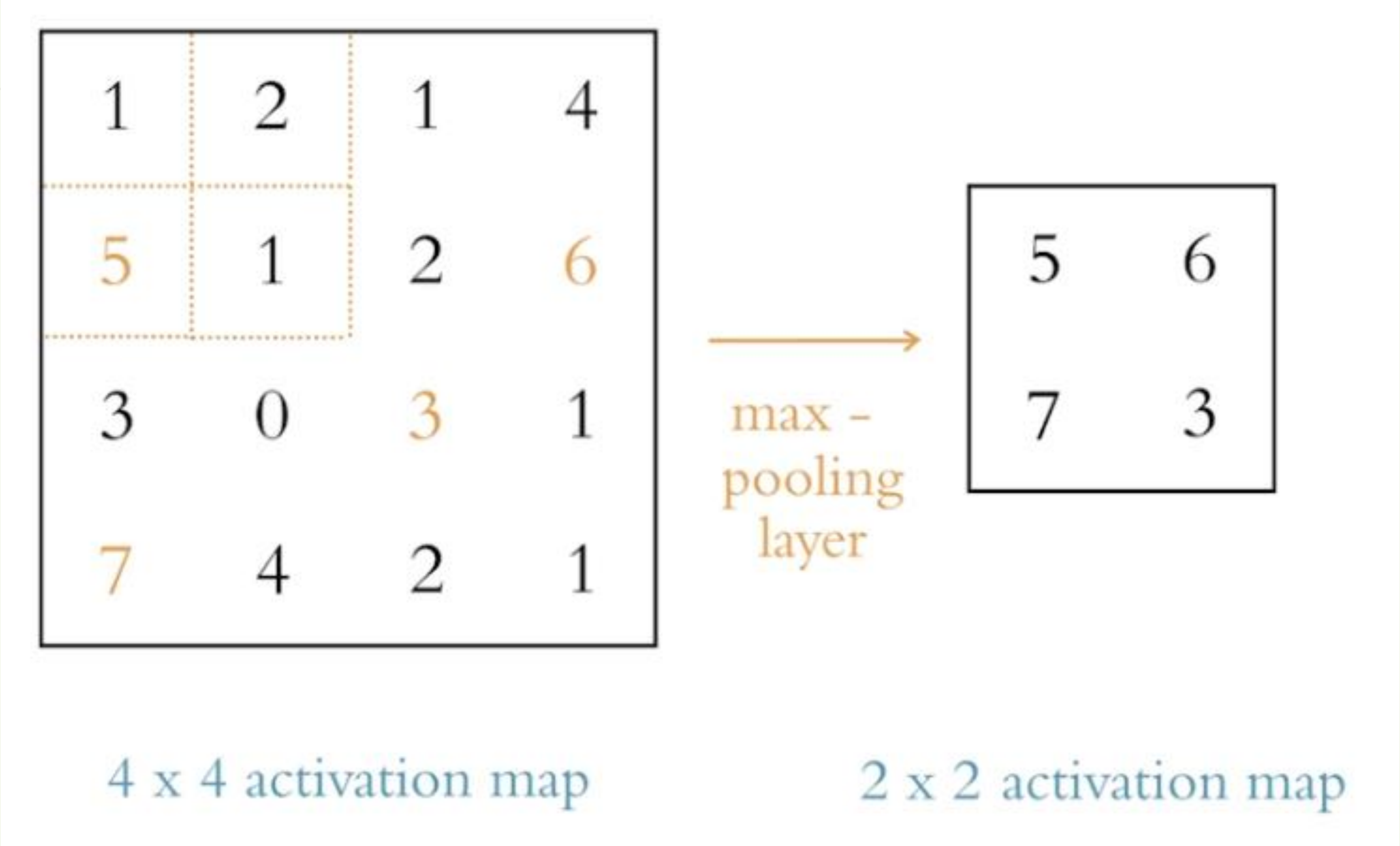
Many layers – building complex image



CNN - AlexNet

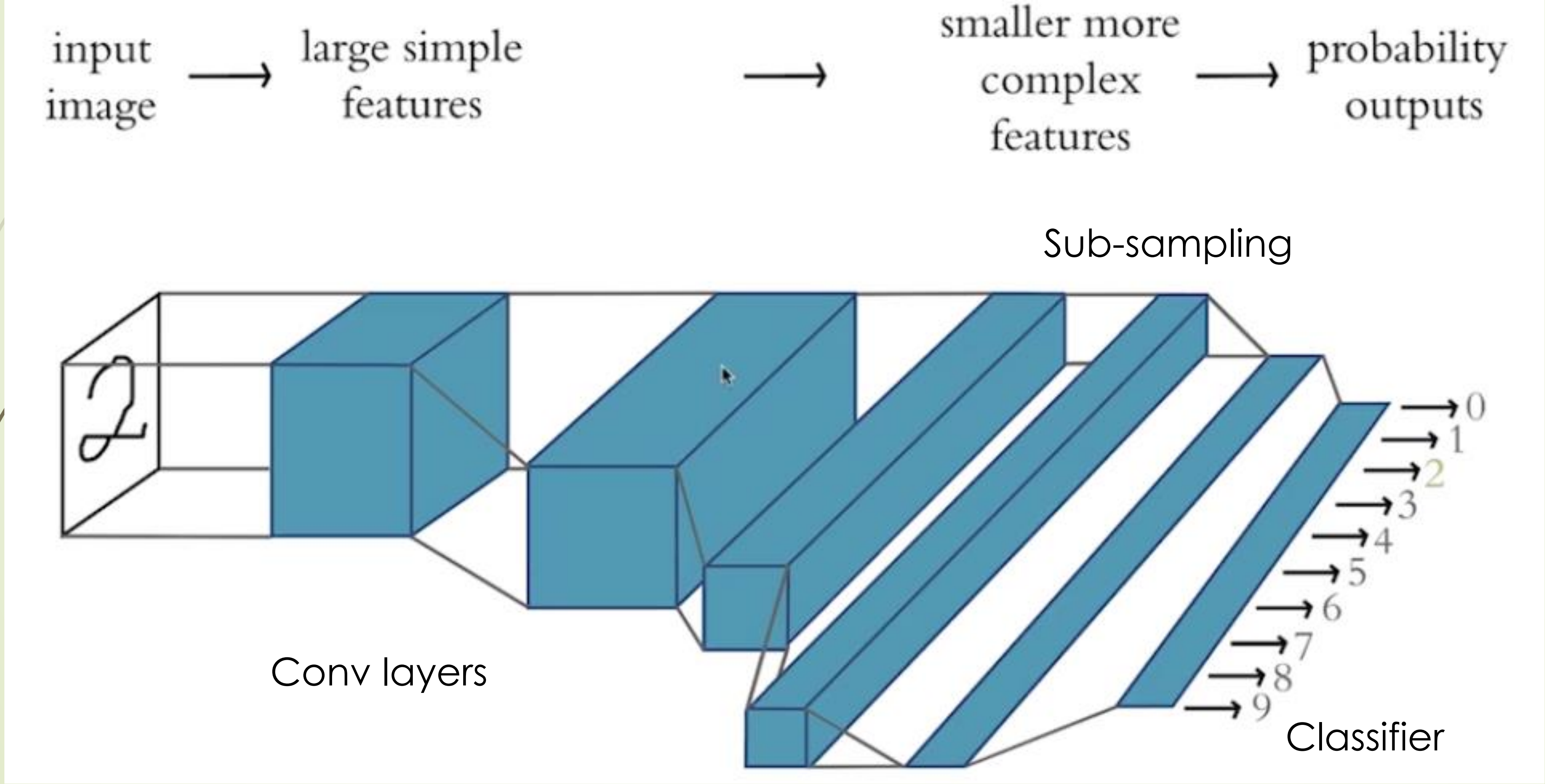


Pooling layer – good partner for CL

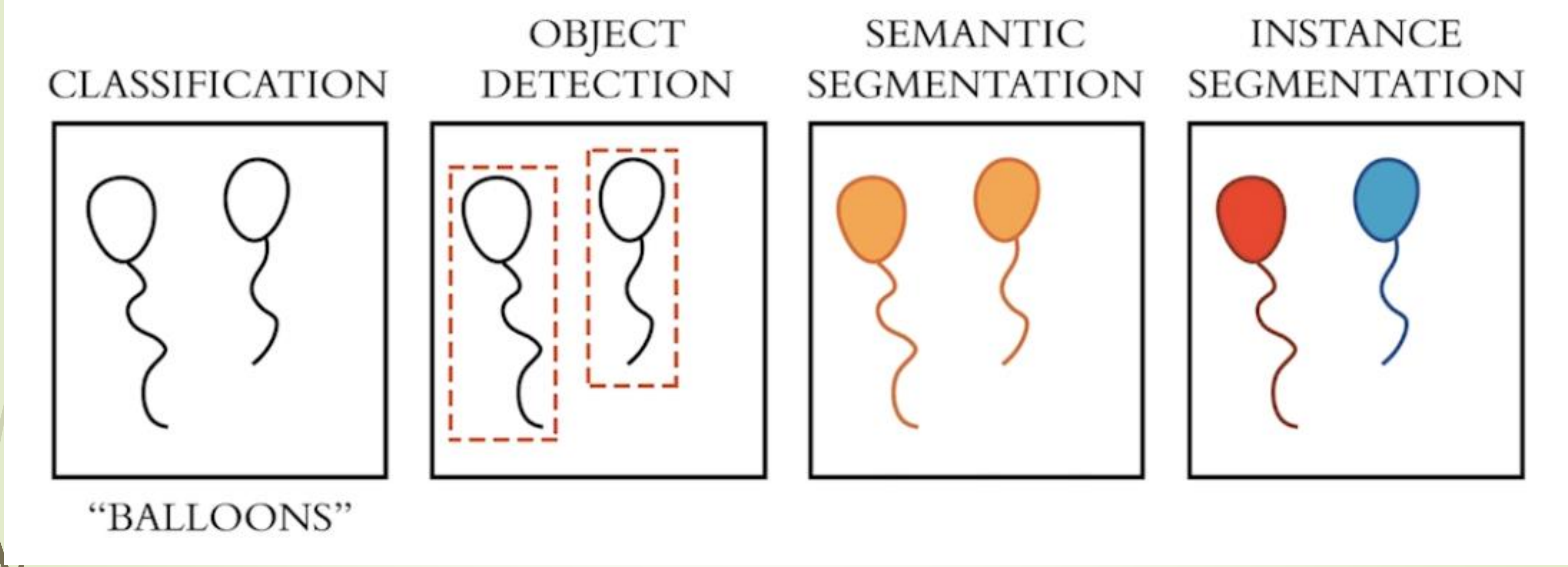


It reduces the size of output activations but preserves the depths (MaxPooling)

Our playground – LeNet-like model



Deep models and thir applications



Capsule network – positional info

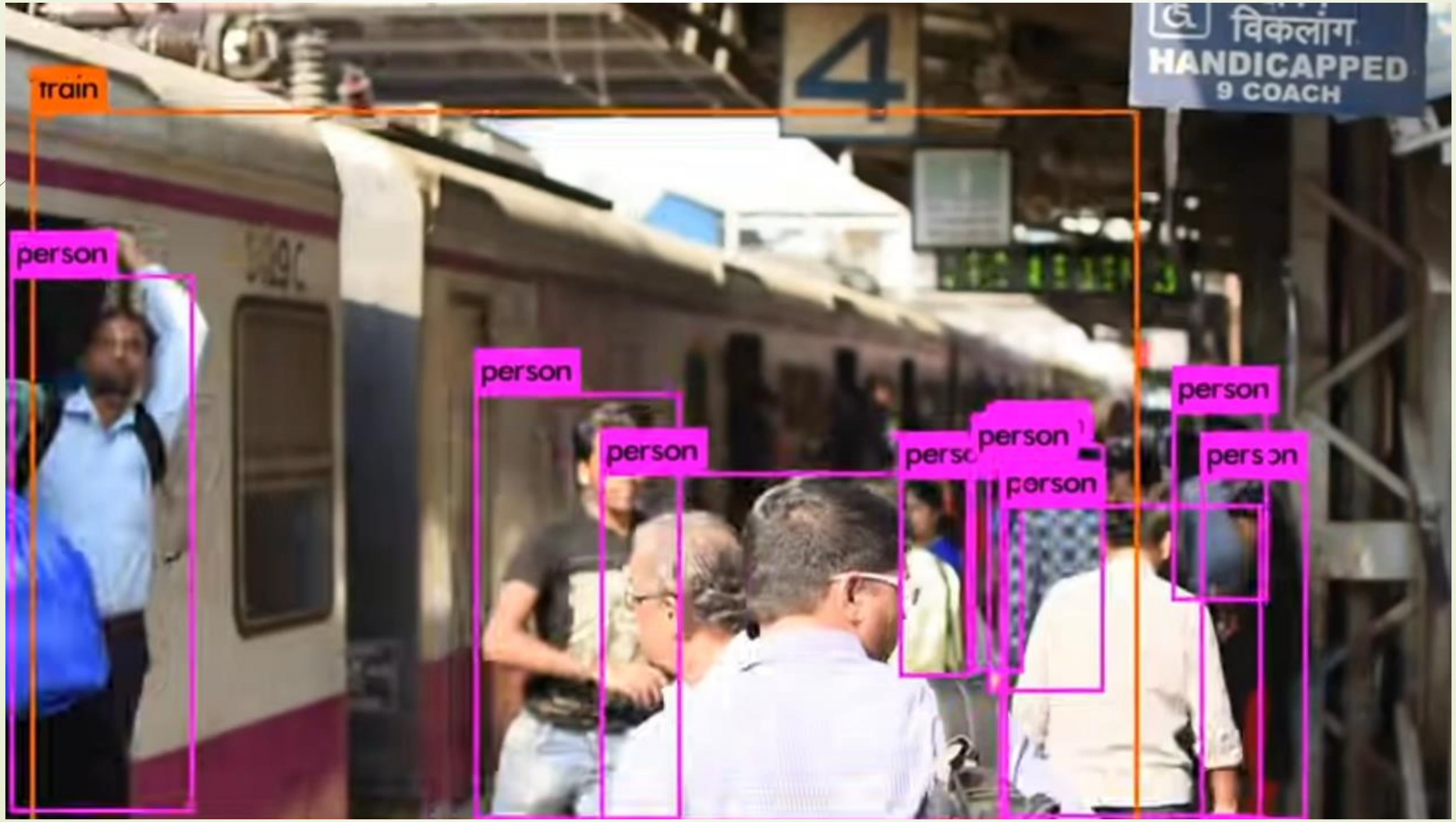
Hyper-parameter tuning

- Perform your model initialisation wisely…
- Define your cost function according to the task
- Set the limit – the goal/target, the random chance is the boundary
- Layers (types, number, sizes)
- Use monitoring tools to check for the over-fitting
- Learning rate tuning
- Batch size tuning

- Automatic tools (Spearmint, Hyperas, Hyperopt)

YOLO – You Only Look Once

- Let's appreciate this for a moment https://arxiv.org/abs/1506.02640



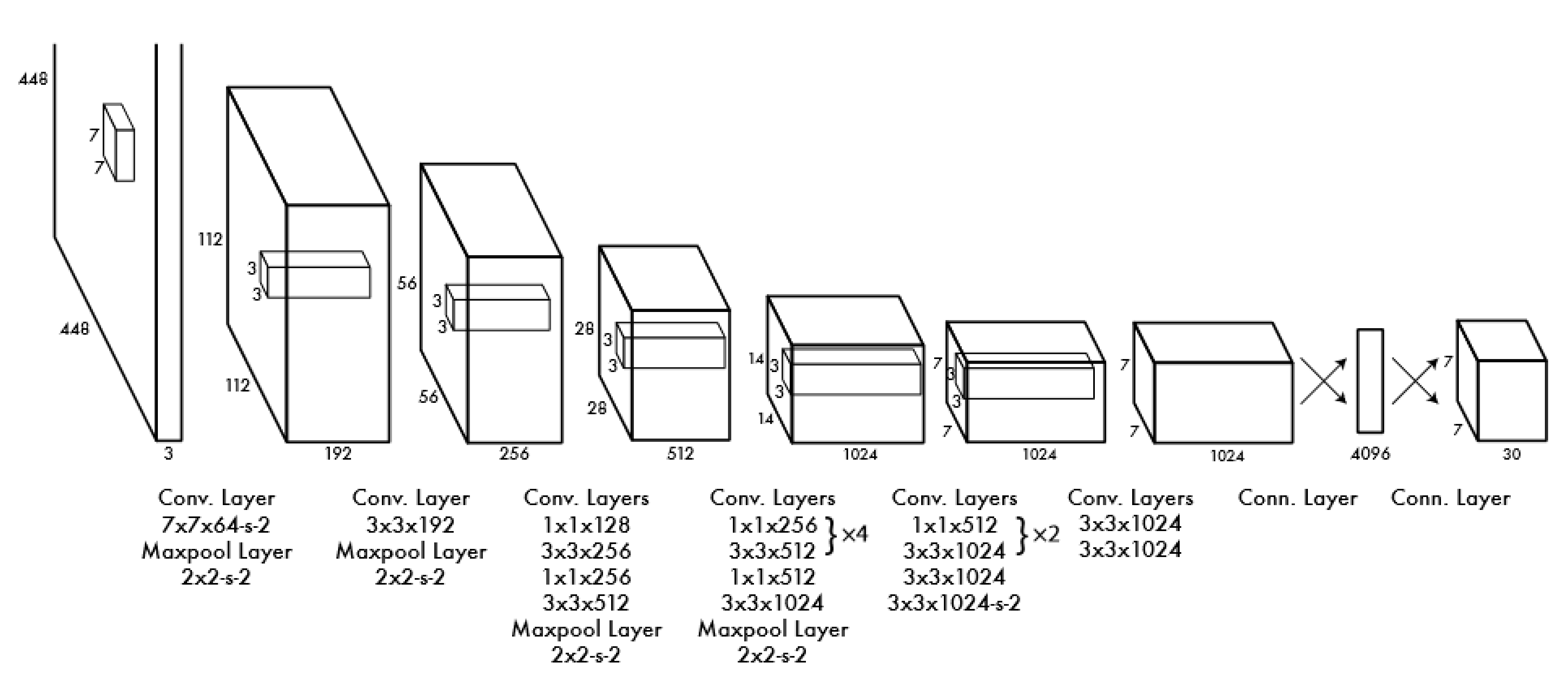
OpenCV – Computer Vision

- For CV we are mostly interested in manipulating of images, OpenCV is one of the most common and loved free piece of software
- It is cross-platform – fairly easy to switch between different Oses
- Usually need to be installed separately
- If you have anaconda on your system you can go for „**pip install opencv-python**"

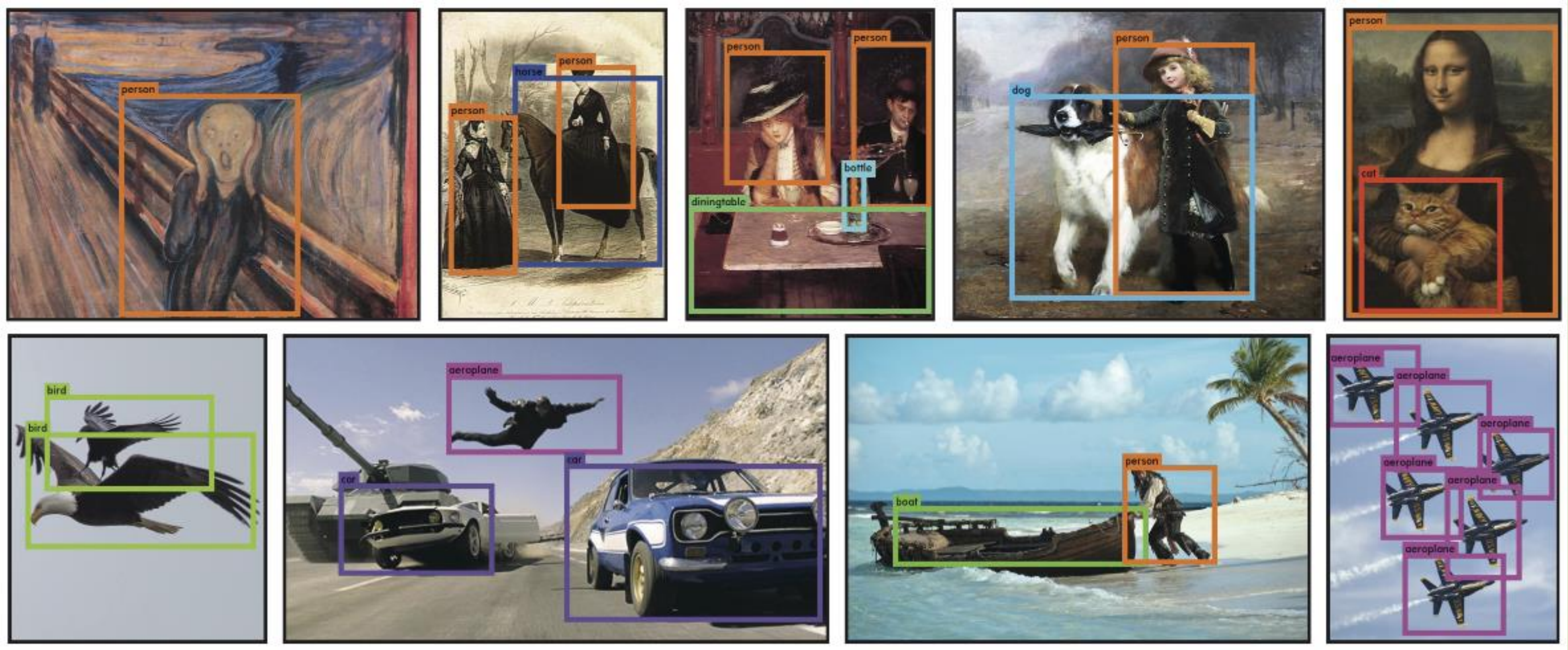
```
#import the libraries
import cv2

#printing the version
print(cv2.__version__)
```

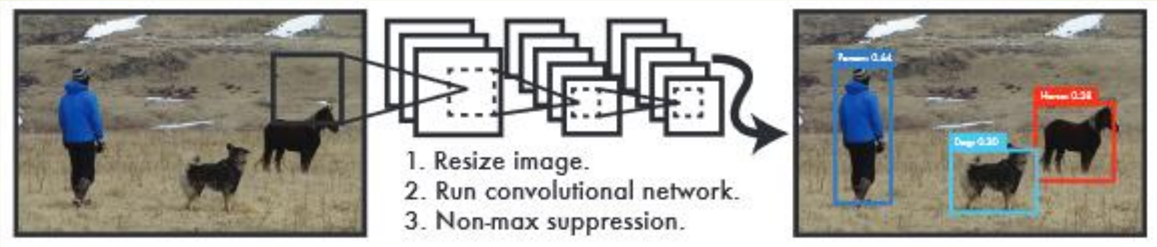
Yolo architecture (2016)

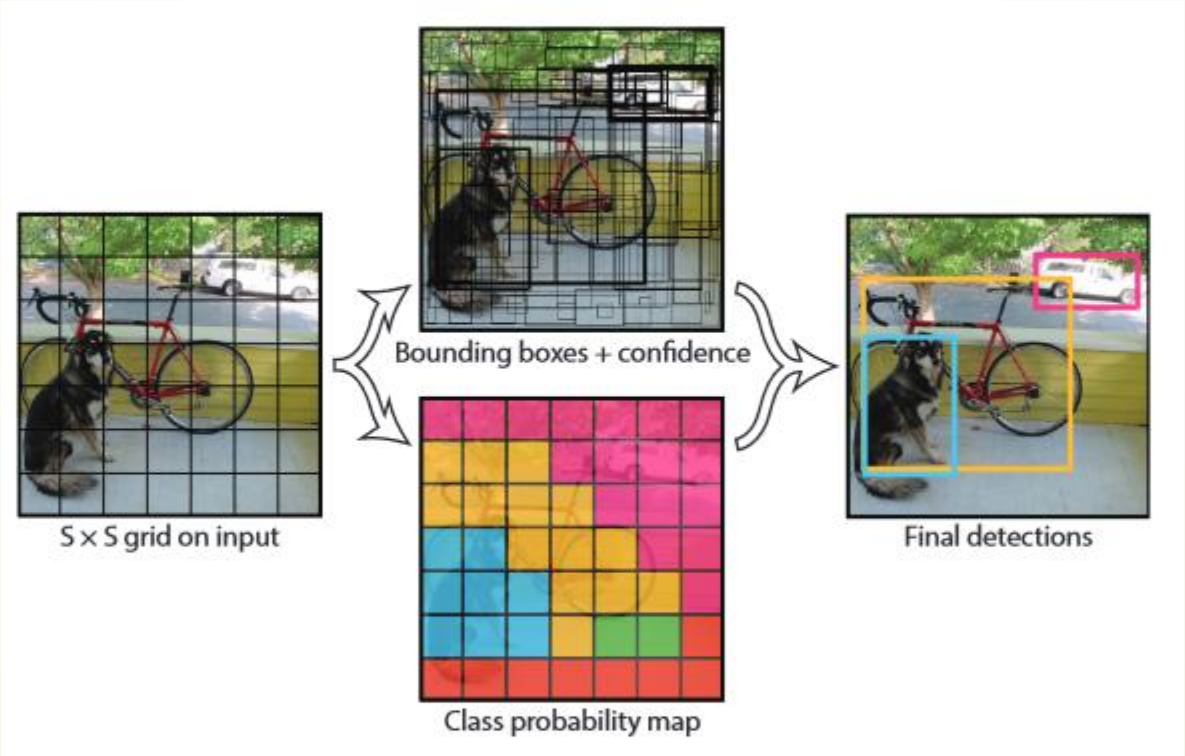


Checking on art datasets



Principle of YOLO

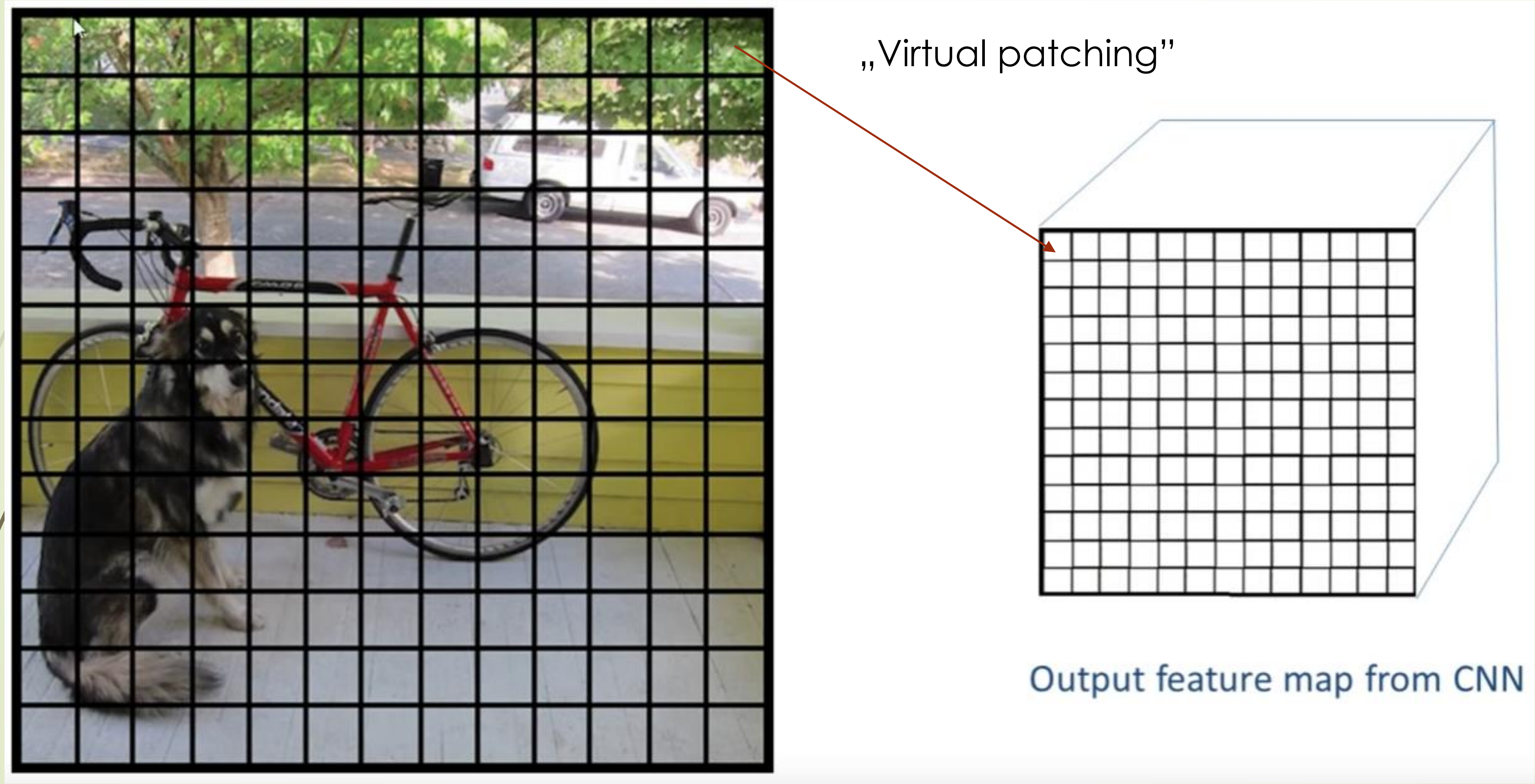




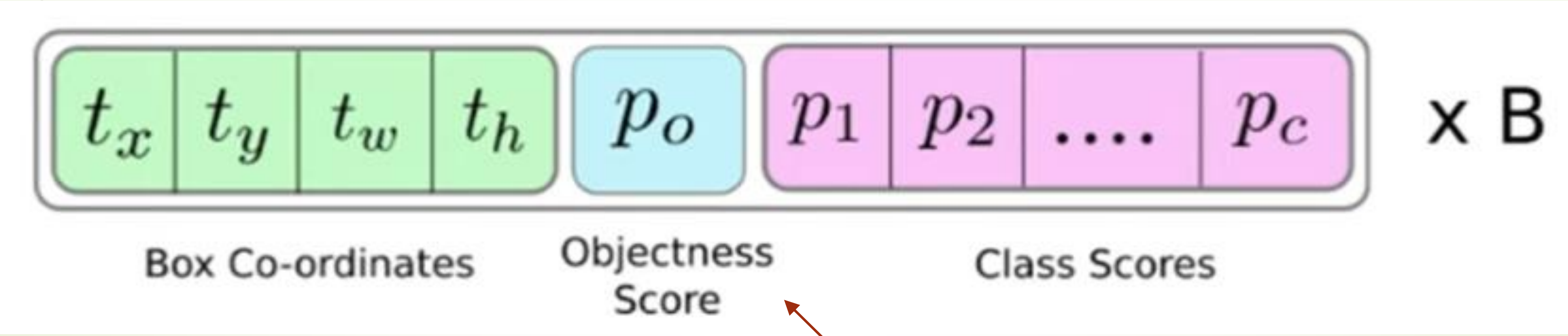
Darknet

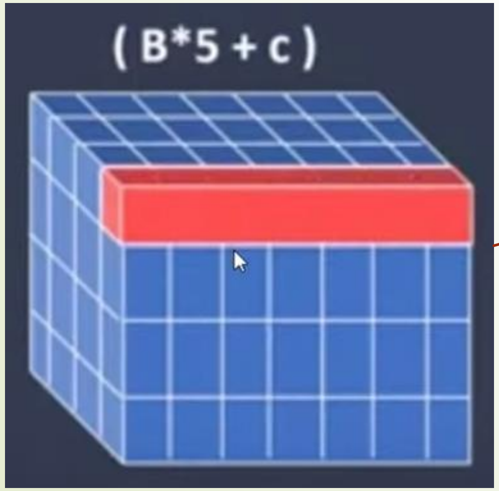
- Original, native environment based on C/CUDA implementation
- We can make it work on colab-like distributed platforms, this is what we are going to do

YOLO innerworkings



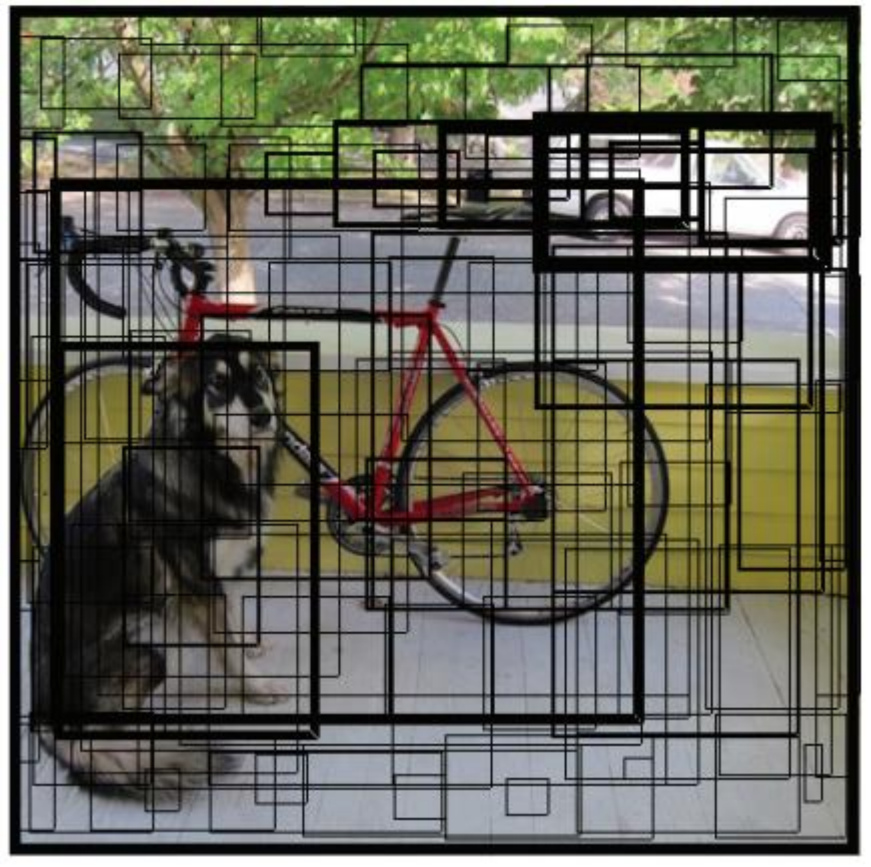
YOLO innerworkings





Formula for a single bounding box for a given patch

Intersection over union



The end